Pg168

4.

1. 源操作数是立即数，应在0xFF前加上”$”.
2. 源操作数长度为16位，与movb的后缀为字节’b’不一致
3. 目的操作数不能是立即数
4. 源操作数长度超过16位，与orw后缀为16位的w不一致
5. 不能用长度为8位的寄存器作为目的操作数地址的值.
6. 源操作数寄存器长度和目的操作数寄存器长度不一，应做操作数拓展
7. 没有%esx寄存器

8)源操作数缺少存放变址值的寄存器

5.

char->int: movsbl %al, (%edx)

int->char: movb %al, (%edx)

int->unsigned: movl %eax, (%edx)

short->int: movswl %ax, (%edx)

unsigned char->unsigned: movzbl %al, (%edx)

char -> unsigned movzbl %al, (%edx)

int -> int movl %eax, (%edx)

6.

1)

xptr: R[ebp]+0x8

yptr: R[ebp]+0x12

zptr: R[ebp]+0x16

2)

void func(int \*xptr, int \*yptr, int \*zptr) {

int tempx = \*xptr;

int tempy = \*yptr;

int tempz = \*zptr;

\*xptr = tempz;

\*yptr = tempx;

\*zptr = tempy;

}

15.

int f1(unsigned x){

int y = 0;

while(x != 0){

y ^= x;

x >>= 1;

}

return y & 0x1;

}

函数f1的返回值是(x^x>>1^x>>2^...)&0x1, 用于检测x二进制编码中’1’的奇偶. 如果是奇数返回1，否则返回0.

17.

unsigned test(char a, unsigned short b, unsigned short c, short\* p){

\*p = a;

return b \* c;

}

22.

M = 5;

N = 7;

25.

1)

node所需的存储空间为16字节

各成员的偏移地址：

p: 0

s.x: 4

s.y: 8

next: 12

2)

void np\_init(struct node\* np){

np->s.x = np->s.y;

np->p = &(np->s.x);

np->next = np;

}

28.

各成员偏移量：

c: 0

d: 4

i: 12

s:16

\*p: 20

l: 24

g: 28

\*v: 36

总大小为40字节

调节顺序结果：

struct {

double d;

long long g;

long l;

int i;

char \*p;

void \*v;

short s;

char c;

}

调整后总共占36个字节;

31.

1)

1 //R[edx] <- M[R[ebp]+0x8] x 送入 $edx

2 //R[ecx] <- M[R[ebp]+0x12] k 送入$ecx

3 //R[esi] <- 255 立即数255送入$esi

4 //R[edi] <- -214783648 将立即数-214783648送入$edi

5 //.L3

6 //R[eax] <- R[edi] 将i送入$eax

7 //R[eax] &= R[edx] ; $eax = i & x

8 //R[esi] ^= R[eax]; $esi = val ^ (i & x)

9 //R[ebx] <- R[ecx] 将k送入$ebx

10 R[edi] >>= R[bl] 将i逻辑右移k位

11 test1 %edi, %edi

12 jne .L3 //if(R[edi] != 0) 跳转到L3

13 //R[eax] <- R[esi]

2)

x存放在%edx中，k存放在%ecx中

val存放在%esi中, i存放在%edi中

3)

val = 255;

i = -2147483648

4)

循环终止条件为i == 0. 循环控制变量i在每次循环中逻辑右移k位.

5)

int lproc(int x, int k){

int val = 255;

int i;

for(int i = -2147483648; i !=; i = (unsigned) i >> k){

val ^= (i & x);

}

return val;

}

33.

1)

n1.ptr: 0

n1.data1: 4

n1.data2: 0

n1.next: 4

2)

大小占8个字节

3)

void chain\_proc(union node \*uptr){

uptr->n2.next->n1.data1 = \*(uptr->n2.next->n1.ptr) - uptr->n2.data2;

}